APPLYING IMAGE MATCHING TO VIDEO ANALYSIS

THESIS

Adam J. Behring, B.S.

AFIT/GCO/ENG/10-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCO/ENG/10-02

APPLYING IMAGE MATCHING TO VIDEO ANALYSIS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Adam J. Behring, B.S.

September 2010

AFIT/GCO/ENG/10-02

APPLYING IMAGE MATCHING TO VIDEO ANALYSIS

Adam J. Behring, B.S.

Approved:

//signed//	
Dr. Gilbert L. Peterson (Chairman)	date
//signed//	
LtCol Michael J. Veth (Member)	date
//signed//	
Dr. Rusty O. Baldwin (Member)	date

AFIT/GCO/ENG/10-02

Abstract

Dealing with the volume of multimedia collected on a daily basis for intelligence gathering and digital forensics investigations requires significant manual analysis. A component of this problem is that a video may be reanalyzed that has already been analyzed. Identifying duplicate video sequences is difficult due to differences in videos of varying quality and size. This research uses a kd-tree structure to increase image matching speed. Keypoints are generated and added to a kd-tree of a large dimensionality (128 dimensions). All of the keypoints for the set of images are used to construct a global kd-tree, which allows nearest neighbor searches and speeds up image matching. The kd-tree performed matching of a 125 image set 1.6 times faster than Scale Invariant Feature Transform (SIFT). Images were matched in the same time as Speeded Up Robust Features (SURF). For a 298 image set, the kd-tree with RANSAC performed 5.5 times faster compared to SIFT and 2.42 times faster than SURF. Without RANSAC the kd-tree performed 6.4 times faster than SIFT and 2.8 times faster than SURF. The order images are compared to the same images of different qualities, did not produce significantly more matches when a higher quality image is compared to a lower quality one or vice versa. Size comparisons varied much more than the quality comparisons, suggesting size has a greater influence on matching than quality.

Table of Contents

List of Figures

List of Tables

APPLYING IMAGE MATCHING TO VIDEO ANALYSIS

I Introduction

Due to volume, intelligence and forensic analysts may reanalyze a video that has been previously analyzed. Performing the identification of duplicate video sequences can be difficult because of different video sizes and qualities. Furthermore, video may have even been analyzed previously by another analyst. With no personal exposure to the prior analysis, the analyst would not even know it was already analyzed.

The National Center for Missing and Exploited Children (NCMEC) keeps lists of hash values of known child pornography images and videos [1]. The lists are disseminated among law enforcement agencies to help identify victims. The Cait series is a series of images often found in child pornography cases. There are 7617 hash values from the 2009 NCMEC list of the Cait series.

Even though having the hash value is strong evidence to show an image is pornographic, every small change in the image causes that hash value to change. Image matching techniques can aid in identification of suspected images by matching image features, which can tolerate small changes, to known child pornography .

Image matching algorithms, such as Scale Invariant Feature Transform (SIFT) [2] and Speeded Up Robust Features (SURF) [3], generate keypoint files from a video or image for later retrieval by an analyst to efficiently search for repeated video sequences.

Possible data structures for storing keypoints are kd-trees [4][5] and Bloom filters [6][7].

Details to consider for a video matching system include the matching algorithm run time, the order the images are compared, and the accuracy of the matching methods.

1.1 Research Goal

The goal of this research is to further the knowledge and methods used in image matching as it pertains to intelligence analysis and forensic investigations.

This research determines if a new video has already been examined or anything new has been introduced that needs to be analyzed. This can reduce the manpower currently taken to analyze new videos that really do not need it. The scope of this research can also be expanded to law enforcement, who examines videos of crimes take from a variety of devices.

- Determine if applying data structures (Bloom filters [6][7], kd-trees [4][5]) benefit the matching process in the form of accuracy and speed
- Determine the impact of image size and lossy compression quality on the matching process

1.2 Sponsor

This research supports the Air Force Research Laboratory (AFRL), Communications Exploitation Branch. AFRL leads the discovery, research and development of information processing techniques by exploiting advanced technologies to intercept, collect, locate, track and process both covert and overt raw multi-sensor data for communications Intelligence.

1.3 Assumptions

All of the images in the first dataset of 125 images are assumed to have come from the same source camera. Likewise all the images from the second dataset of 298 came from the same source video. There were no comparisons or testing done between the two sources. The 102 keypoints selected for the keypoint reduction was based on [8]. This number was chosen through trial and error. The autopano-sift software [5] was designed for panoramic stitching and not for image matching in general.

1.4 Organization

Chapter II provides background information on image feature generation algorithms (SIFT, SURF, HOG), image matching techniques, hashing techniques and data structures (Bloom filter, KD-tree). Chapter III presents keypoint reduction methods, KD-tree generation and RANSAC model matching. Chapter IV describes the data sets, tests used and the results from the testing. This chapter also discussed the accuracy of the algorithms, computation time and quality and size variation comparisons. The final chapter discusses conclusions and future work.

II Related Work

This section reviews previous work related to image and video matching. It discusses matching algorithms and the extraction of high-resolution stills from video and critical video quality. Hashing methods, used to quickly determine if matches exist in the database of previously examined images/videos, are reviewed and discussed.

Image features generated by the Scale-Invariant Feature Transform (SIFT) and the Speeded-Up Robust Features (SURF) algorithms are discussed. Once the features are generated they will be added to a data structure. Storing the features in a data structure allows for faster comparisons. The data structures discussed include Bloom filters [7] and kd-trees [5]. Finally, previous research on image and video matching is discussed. Including extraction high-resolution stills from videos and the critical video quality.

2.1 Image Features

Image features are pieces of information such as edges and contrast extracted from images for computer vision applications. Features are calculated differently depending on the algorithm.

2.1.1 Scale-invariant feature transform (SIFT)

The SIFT algorithm [9] performs image recognition by transforming images into local feature vectors called keypoints. Keypoints are invariant to translation, scaling, and rotation, and partially invariant to illumination changes.

The algorithm generates a large number of keypoints, roughly 1000 for a 512x512 pixel image. Given such a large number of distinctive keypoints, the algorithm can match scenes and objects with a high probability [2]. The algorithm has four major stages.

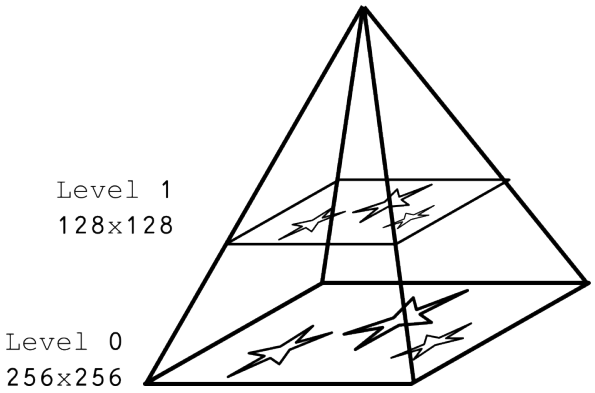

Figure 1: Image pyramid example.

The first stage looks for locations that are the maximum or minimum of the difference-of-Gaussian function. This function enhances a grayscale image, the maxima and minima of which are used to generate a feature vector. An image pyramid is built to determine maxima and minima pixels. The pyramid consists of the smoothed and resampled image at each level. An example of this is in Figure 1. Since the 2D Gaussian function is separable, its convolution with the input image can be efficiently computed by applying two passes of the 1D Gaussian function, below, in the horizontal and vertical directions [2]

$$g(x) = \frac{1}{\sqrt{5\pi}\sigma} e^{-x^2/2\sigma^2}. \quad (1)$$

The first image, A, is computed using the Gaussian function with $\sigma = \sqrt{2}$. The Gaussian function is applied again to produce a smoothed new image, B. The difference-of-Gaussian function is found by subtracting image B from image A. The pixels in the difference-of-Gaussian image are compared to their 8 immediate neighbors and their 9 neighbors in each neighboring pyramid scale level. If the pixel is a maxima or minima compared to its 26 neighboring pixels then it is selected as a potential keypoint.

Next, keypoints are chosen based on measures of their stability, that is a model fit to their location and scales [2]. To determine if the potential keypoint is stable enough each scale level of image A is processed to extract the gradients and orientations using

$$\begin{aligned} M_{ij} &= \sqrt{(A_{ij} - A_{i+1,j})^2 + (A_{ij} - A_{i,j+1})^2} \\ R_{ij} &= atan2(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{ij}) \end{aligned} \tag{2}$$

where A_{ij} is the i, j^{th} pixel, M_{ij} is the gradient magnitude, and R_{ij} is the orientation. The image is subjected to affine projections, contrast and brightness changes, and addition of noise. The keypoints in the original image can be predicted in the transformed images from the knowledge of the transform parameters. The stability of the keypoints to image transformation is determined by the percentage of keypoints in the first image and with matching keypoints in the transformed image. These keypoints are in areas of high variation. Orientations are assigned to the keypoints based on the gradient of the image. Finally, gradients around the keypoints are

measured and transformed into a representation that can specify shape distortions and changes in illumination [2].

To match two images, the keys from each image are compared to every key from the other image to find matches. Normally, a single keypoint is matched to a large set of features. If the image is cluttered, many of the matches will be false positives. By identifying subsets of keypoints that correspond between images and their location, scale, and orientation, correct matches can be found [2]. Each subset of 3 or more corresponding features of an object and its orientation is tested further. A least-squares estimate is made to approximate the object's pose. Any other keypoints consistent with this pose are identified, and outliers are discarded. Finally, a detailed computation is made of the probability that a particular set of keypoints indicates the presence of an object. Object matches that pass all these tests are correct with high confidence.

Images featuring an indoor environment can increase computational time while decreasing the complexity of the SIFT keypoints [10]. Images from indoors include walls and corners. Floors and ceilings, because of their low contrast uniform texture, rarely generate many SIFT keypoints. Assuming that the indoor images experience little view point rotation then the keypoints from the walls will not rotate. The algorithm can be sped up by omitting three of its steps, the calculation of keypoint orientation, the generation of additional keypoints at locations with multiple dominant orientation and the alignment of the keypoint descriptor to the keypoints orientation [10].

The SIFT algorithm has been expanded using principal component analysis (PCA) [11]. The PCA-SIFT algorithm modifies the fourth stage in the SIFT algorithm, resulting in a key point representation that is simpler, more compact, faster and more accurate [11]. A 41x41 patch is extracted at the given scale, centered over the keypoint and rotated to align its dominant orientation to a canonical direction. The algorithm then computes an eigenspace to express the gradient images of the patches. The gradient is determined for a given patch and a compact feature vector is derived using the eigenspace and the gradient image vector. The keypoint is much smaller than the normal SIFT keypoint.

Comparison between PCA-SIFT and SIFT used a dataset of 30 images containing 10 household objects. Each object was photographed from three different angles. An image was then compared to the remaining set using PCA-SIFT and SIFT. Two points were awarded to the algorithm if the two other images of the corresponding object were returned in the top three positions. If only one image was returned in the top three then 1 point was awarded, it was awarded no points for all other cases [11]. The number points awarded was divided by 60 (the total number of correct matches) calculate the accuracy. Using this measure SIFT had an accuracy of 43% and PCA-SIFT had 68% for the dataset used. The PCA-SIFT algorithm provided better matching accuracy and speed for controlled and real-world images [11].

2.1.1.1 SIFT Keypoint Reduction

The SIFT algorithm generates a large number of keypoints per images. Reducing the number of keypoints increases the speed in the matching process and

reduces the storage requirements. The method of reduction keeps the strongest keypoints to ensure the best accuracy in the matching process. Keypoint reduction is achieved with the following Mahalanobis distance function to ensure a good keypoint spread [8]. Keypoint reduction is an iterative process starting with two points selected based upon the scale of the detected keypoints. The keypoints are selected by evaluating each point (x_i, y_i) using $W_1 D_{mahal}(x_i, y_i) + W_2 \sigma(x_i, y_i)$ to get the highest value where $\sigma(x_i, y_i)$ is the scale, D_{mahal} is the Mahalanobis distance

$$D_{mahal}(x_i, y_i) = \sqrt{(x_i - y_i)^T M (x_i, y_i)} \tag{3}$$

at point (x_i, y_i), M is the covariance matrix, W_1 is the weighting of the Mahalanobis function and W_2 is the weighting of the scale of the keypoint [8]. Keypoint selection continues until the desired number of keypoints are found. In the case of [8], 102 keypoints are selected. This ensures that the selection of keypoints are spread uniformly and are still strong choices for matching.

2.1.1.2 SIFT Match Comparisons

Two quality checks are performed to reduce poor keypoint matches during post processing [8]. Both import the match points from an output file and the match points (x_1, y_1) and (x_2, y_2) are converted into lines representing the match lines. The equation $y = mx+b$ represents that line with the slope

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{4}$$

and the y-intercept, $b=y_1-(mx_2)$. The intersections of all the match lines are calculated. If the slopes, m_1 and m_2, of the two lines are equal they are considered parallel. Otherwise x is calculated using

$$x=\frac{-(b_1-b_2)}{m_1-m_2}. \tag{5}$$

Using (5) the two possible values of y are calculated from $y_1 = m_1x + b_1$ and $y_2 = m_2 x + b_1$. If the values of y_1 and y_2 are equal then the point (x, y_1) is added to the intersection list.

Both methods use intersections to determine poor matches that should be removed. The first method determines bad matches by finding lines intersecting other match lines in the frame. Figure 2 shows an example of this. A diagonal match line intersects the other match lines within the image frame. The match is considered poor and is excluded. Lines that are parallel are kept as good matches, as shown in Figure 7. The second method uses the same initial steps and computes a mean intersect point from the average x and average y for all the intersect points (x, y)

$$\bar{x}=\sum_{i=0}^{N}\frac{x_i}{N}, \text{and} \tag{6}$$

$$\bar{y}=\sum_{i=0}^{N}\frac{y_i}{N}. \tag{7}$$

The average distance from the mean

$$\bar{d} = \sum_{i=0}^{N} \frac{\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}}{N} \tag{8}$$

and standard deviation

$$\sigma = \sqrt{\sum_{i=0}^{N} \frac{(d_i - \bar{d})^2}{N}} \tag{9}$$

of the intersect point are calculated using the average intersect point $(\bar{x}, \bar{y})$.

The distance of the intersect points of each line from the average intersect point $(\bar{x}, \bar{y})$ is checked against σ. If the sum of the distances greater than σ is 90% or more of the total number of intersect points associated with the line [8], the line is marked as bad.

2.1.2 Speeded-Up Robust Features (SURF)

The SURF algorithm [3] operates in a similar fashion as the SIFT algorithm. It first finds keypoints or interest points [3]. using a Hessian Matrix. The interest points are specified by Haar wavelet responses. A 4x4 square is laid over the interest point and the Haar response is computed for each square. The responses are summed up over each sub-region and form the first set of entries in the feature vector [3]. The two image vectors are compared to find matches. The matching process can be improved by including the sign of the Laplacian (i.e., the trace of the Hessian matrix) [3] to distinguish between bright blobs on dark backgrounds and the reverse. Only features with the same type of contrast are compared in the matching stage. The interest points

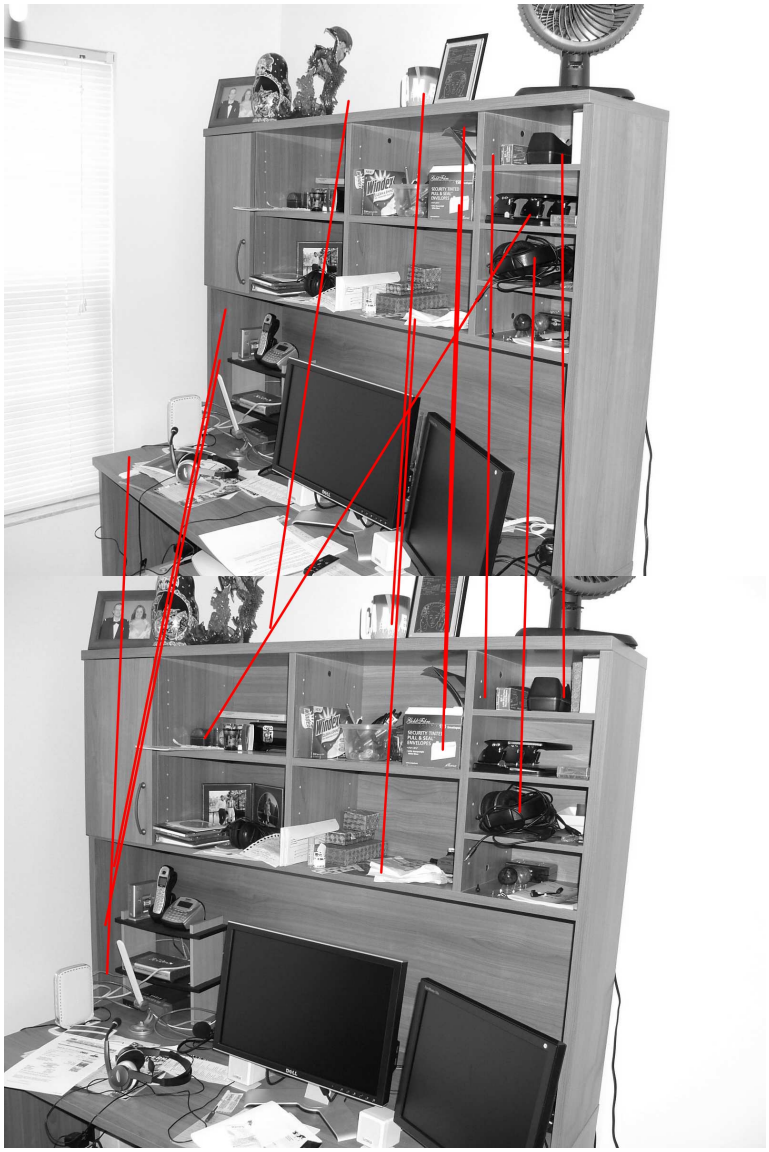

Figure 2: Quality check example [8].

of this algorithm are scale and rotation-invariant, like the SIFT algorithm.

2.1.3 Histogram of Oriented Gradients (HOG)

The Histogram of Oriented Gradients (HOG) [12] are similar to the keypoints used in the SIFT algorithm, but they are "computed on a dense grid of uniformly spaced cells and they use overlapping local context normalizations for improved

performance" [12]. This algorithm was originally developed for pedestrian detection. That is, mostly visible people in a relatively upright position. This algorithm has shown good results applied to infrared images [13] and video streams [14].

The first step in this process is to compute the gradients of the image. The gradient is determined by applying two 1-dimensional filters, [-1 0 1] and $[-1\ 0\ 1]^T$, to the horizontal and vertical, respectively. The pixels are organized into square-like spatial regions called cells. The cell has a predefined size in pixels. Each pixel calculates a weighted vote for an edge orientation histogram channel based on the orientation of the gradient [12]. The gradients can be signed or unsigned, meaning the contrast information that the signs provide is unimportant to the algorithm. Cells are organized into blocks in a sliding fashion. Which causes the blocks to overlap one another. The overlapping allows the cells to contribute several components to the final descriptor vector and significantly improves the performance [12].

Cells are grouped together into larger, spatially connected blocks to normalize the contrast. Triggs and Dalal [12] explored four normalization schemes, all of which provided improvement over the non-normalized data. Let v be the non-normalized descriptor vector, $||v||k$ be its k-norm for k=1,2, and ϵ be a small constant. The schemes are then:

$$L2-norm: f = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}},$$
$$L1-norm: f = \frac{v}{\|v\|_1 + \epsilon}, \quad (10)$$
$$L1-sqrt: f = \sqrt{\frac{v}{(\|v\|_1 + \epsilon)}}.$$

The fourth scheme, *L2-Hys,* is *L2-norm* with clipping which limits the maximum values of *v* to 0.2 and renormalizes.

The most frequent application of HOG descriptors is to identify standing pedestrians in an image [13].

2.2 High-Resolution Stills

While these feature generation algorithms operate on still images they can also be applied to the intra-coded frames (I-frames) of videos in the MPEG format. I-frames are key frames in an MPEG video, and are full frames, and can be treated similarly to an image. The other frames are bi-directionally predictive-coded frames (B-frames) and predictive frames (P-frames) but they do not contain as much information and are, consequently, not suitable for video matching [15]. The MPEG format uses a group of pictures (GOP) composed of the three different frames, shown in Figure 3.

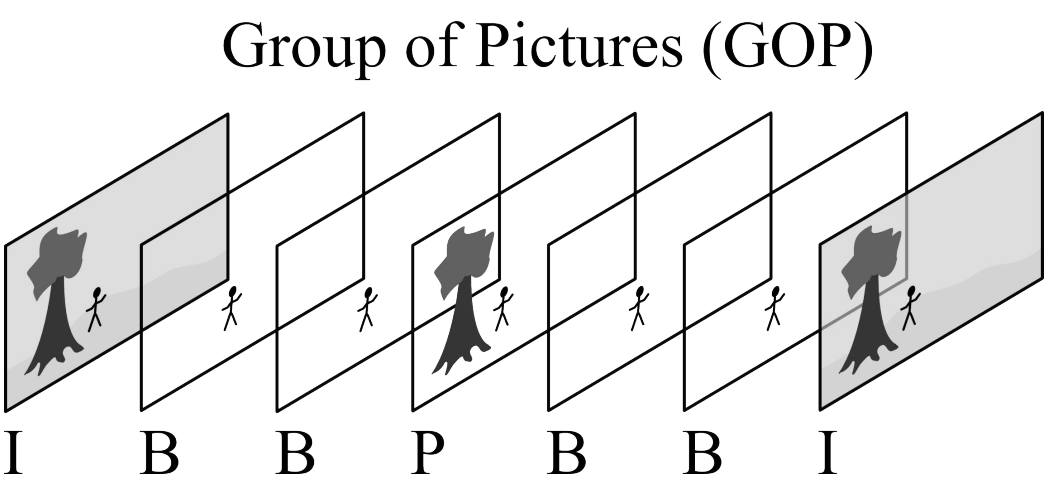

Figure 3: Group of Pictures structure.

Matches are made between videos by extracting the I-frames from the GOP and applying a matching algorithm. There has been some research to enhance the I-frames taken from videos by integrating the neighboring P-frames and B-frames [15]. The improvement process uses information from the surrounding frames to estimate the high-resolution information in the I-frame. Enhancing lower quality video using this method may help produce more accurate key points for the matching algorithms. With more keypoints more reliable matches can be established between higher quality and lower quality videos. Introducing more than four or five frames into the high-resolution enhancement algorithm, however, reduces the quality of the image [15].

The frames directly predicting an I-frame, that is neighboring frames up to the first P-frame in a GOP down to and excluding the P-frame of the previous GOP, are used to compute a high resolution video still. The frames that contribute most to the high resolution still can be seen in Figure 4. If two frames were used to estimate a high-resolution still, the I-frame and the first provide the most information. If three

frames were used then the I-frame, P-frame, and B-frame immediately before the I-frame should be used.

Once the frames and corresponding motion fields are decoded, each high-resolution motion field is computed by up-sampling using a cubic B-spline interpolation [16]. Block matching determines the displacement of a pixel block within a search region between the two frames and is applied to the two up-sampled frames. Any inaccurate motion vectors are detected and the corresponding pixel removed from the video observation model. The high-resolution video still is then estimated using the decoded frames and subpixel motion fields.

2.3 Critical Video Quality

Given each video has multiple I-frames and each I-frame has thousands of keypoints, a system could exhaust available space to store such information. There is, however, a "sweet spot" between video quality and accuracy that has been dubbed critical video quality [17]. This has been applied to facial matching and tracking algorithms, but can also be applied to video matching. During testing facial detection, bandwidth usage was reduced up to 29 fold [17] using a prototype surveillance system operating in two modes. When the surveillance system is not detecting faces it streams low quality video. When a face is detected the system raises the quality of the video. Thus, the bit rate is reduced between surveillance cameras and the monitoring stations. This can also be applied to video matching by finding the critical video quality to reduce storage space without compromising the accuracy of the matching algorithm. High-resolution stills can improve this accuracy by increasing quality and

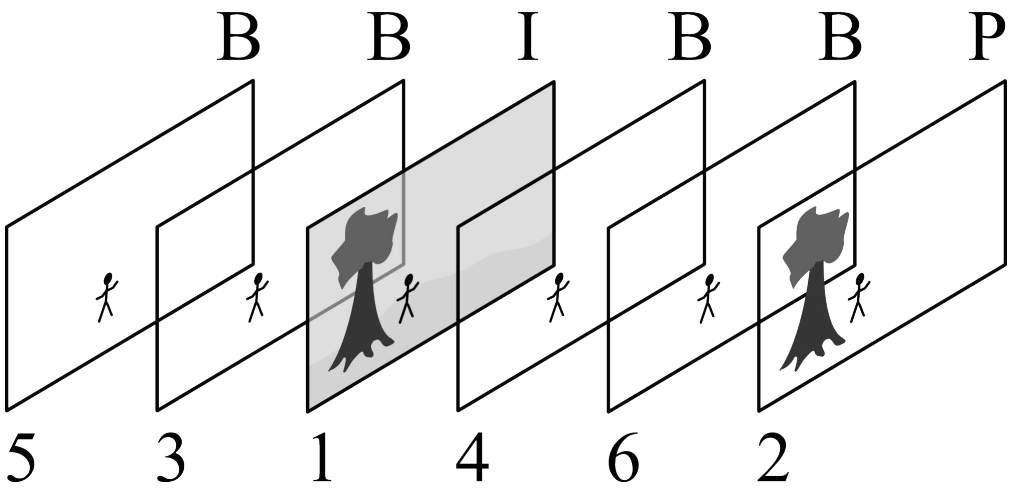

Figure 4: GOP Enhancement order.

thus keypoints.

2.4 Hash Techniques

Hash functions are mathematical functions that take large sets of data as input and output a smaller set of data. An index to the inputs location is often placed in a table or array. Hash functions enable fast table lookups to detect similar or duplicate entries in a large file. Hash functions must be deterministic and provide uniformity across its output range.

Some hash tables use a single hash function. The input and the resulting value is the index into the table where that input is stored. The tables are generally linked lists and provide a fast way to find entries in large datasets like dictionaries or word lists.

Cryptographic hash functions are used for verification and authentication [18]. These hash functions have several notable characteristics. It is difficult to determine

the original input given the hash digest as any change in the input results in a large change in the digest and it is extremely improbable to find two different inputs that result in the same digest. Regardless of the original input length the message digest will be a fixed length. These desirable properties lead to cryptographic hash functions for security tasks. Passwords can be stored securely as their digests. Using a public key encryption scheme data can be signed to provide authentication or privacy.

Geometric hashing is used in computer vision tasks [7]. First a set of affine features are extracted from an image using feature algorithms like SURF or SIFT. The multidimensional hash value is normalized for the space defined by the set of features.

Bloom filters use multiple hash functions (specific or arbitrary) to determine and construct a membership table [7]. Inputs are evaluated by the hash functions, the outputs of which is used as indices in the hash table. The hash output indices are set to 1 to signify the input is a member of the Bloom filter. A query can be made to determine if the query is possibly in the table or not. The query goes through the same hashing process and the output indices are checked if they are set to 1. If all of the indices are set to 1 then the query may be a member of the Bloom filter, but if any of the indices are set to 0 then the query is not a member of the Bloom filter. A Bloom filter does not allow the removal of elements from the filter, only additions. Furthermore the Bloom filter is a probabilistic hash. That is, there may be false positives, which correlates to matches of similar but not exactly the same features.

2.4.1 Bloom Filter

A Bloom filter is a simple and efficient randomized data structure that represents a set and supports membership queries [6]. Bloom filters can give false positives and the probability of a false positive increase with the size of the data structure. Even with false positives, the Bloom filter's simplicity, performance and space efficiency ensure wide use. A Bloom filter is a bit array with all bits initially set to zero. Multiple hash functions are used for robustness and reduce the likelihood of collisions in the array. Elements can be added to the set and the set can be queried to determine if an element is part of the set.

For example, suppose the element *x* is being added to a Bloom filter as seen on the left side of Figure 5. Three different hash functions are applied to *x*, giving the indices of 3, 5, and 9. The element at those indices in the Bloom filter are set to 1. The right side shows *y* being added to the Bloom filter. The three hash functions, when given *y* as an input, produce the indices 1, 5, 8. These indices are also set to 1 in the array.

In a similar fashion it can be determined if an element is contained in the Bloom filter. The query is hashed and if any of the locations in the bit array are zero then it is not part of the set. Figure 6 shows how a false positive can occur. Given the same Bloom filter from Figure 5 containing elements *x* and *y*, a query of *w* with indices of 1, 5, and 9 will show that each index from query *w* is one. This however is a false positive because the indices that are one included in the filter due to elements *x* and *y*. Elements cannot be directly removed from the Bloom filter. However,

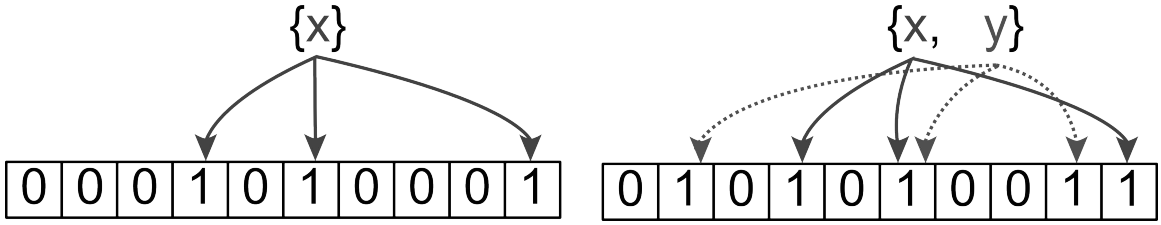

Figure 5: Bloom filter insertion example.

another Bloom filter can be constructed to contain only the elements one wishes to remove.

The Bloom filter performance can be improved using the technique in [6]. Bloom filters are simple data structures and any improvement in Bloom filter operations, translates into an application speed up as well. By modifying the Bloom filter's construction and evaluation of it's pseudorandom hash functions a computational reduction can be achieved.

For example, two hash functions, $h_1(x)$ and $h_2(x)$, can simulate many hash functions of the form $g_i(x) = h_1(x) + ih_2(x)$, with i being in the range of 0 to $k-1$ to generate k hash functions [6]. Using two hash functions in this way does not increase the false positive rate any more than that of a normal Bloom filter.

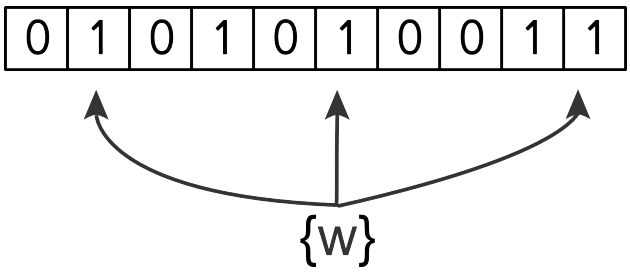

Figure 6: Bloom filter collision example.

2.5 Image Matching

Image matching techniques usually involve acquiring keypoint features from the images in questions and comparing them. These techniques apply to facial recognition, pattern recognition, background matching [8], digital video forensics [19] and ballistics matching [20].

Casings from bullets show distinguishing characteristics that tie it to a particular firearm. The firing, feeding and ejection mechanisms all leave their marks on the casing [20]. The features are calculated from the test image and indexed into descriptors that are searchable. Preprocessing includes converting the image to grayscale and isolating the center, circular region of the breechface, or firing pin. A derivation [21] of the Kanade Lucas Tomasi equation

$$\epsilon = \int \int_{W} \left[J(\boldsymbol{x} + \frac{d}{2}) - I(\boldsymbol{x} - \frac{d}{2}) \right]^2 w(\boldsymbol{x}) dx \tag{11}$$

is used to calculate the dissimilarity, ϵ, where W represents two windows, one with image I and another image J, $\boldsymbol{x} = [x, y]^{\mathrm{T}}$, the displacement $d = [d_x, d_y]^T$ and the weighing function $w(x)$ is usually set to 1. Good features are located by examining the minimum eigenvalue of each 2x2 gradient matrix [20]. Prominent features are selected and stored in a database for comparison to other images. The number of matching points between the two images measures the similarity.

Feature matching can determine distinct locations from a set of images [8] which results in a subset of images with the same background that are useful in

determining suspect, witnesses or other individuals at a crime scene. This method generates keypoints to compare two images.

The SIFT algorithm achieved a 81.6% accuracy using unreduced keypoints and a matching threshold of 140. The accuracy of 81.1% was achieved with reduced keypoints and a matching threshold of 6 [8]. The SURF algorithm achieved a 78.3% accuracy when using unreduced keypoints and a matching threshold of between 1351 and 1363. The accuracy of 79.6% was achieved using reduced keypoints and a matching threshold of 57. The accuracy of 80.8% was achieved with the inclusion of the comparison test [8]. The SIFT and SURF algorithms performed the best with no statistically significant change in accuracy [8]. This research however, did not include the PCA-SIFT algorithm in testing.

Other methods are applied to fingerprints and facial recognition [22]. Fingerprint recognition uses minutiae extraction and pattern recognition. The ridges are reduced to one pixel width and the minutiae along that ridge are saved. The two patterns are spatially aligned and an edit distance is computed. This shows the minimum cost edit operation to transform one minutiae pattern into the other [22]. The facial recognition system uses a feature extraction method with eigenfaces and fisherfaces. This research also provides a unifying approach for the analysis of any type of evidence [22] that can be applied to video and image matching for intelligence and forensics.

2.6 KD-Tree

First discussed in terms of image and video [4], a kd-tree is a data structure

with points in a k-dimensional space. Each non-leaf node can be thought of as a hyperplane that splits the dimensional space into subspaces. There are number of methods to determine were the split should occur. The standard split divides the dimensional space based on which data points have the maximum spread [4]. The splitting value is the coordinate median of the points in that dimension. The midpoint splitting method splits the dimensional space through the center and bisects the longest side of the dimensional space [23]. The sliding-midpoint method attempts a midpoint split. If all the data points are on one side of the hyperplane then the plane slides toward the points until it encounters the first of the data points. This leaves the single data point as a leaf and the splitting recurses on the remaining points [23]. The points on either side of the hyperplane form the left and right subtrees. The kd-tree data structure supports adding/removing elements, balancing and searching.

Searching is done using a Nearest Neighbor algorithm. The algorithm starts at the root node and moves left or right down the tree depending on whether the value is greater than or less than the current node. The algorithm works its way to a leaf node and saves that value as the current best or current nearest neighbor. The algorithm proceeds recursively through the tree comparing the current node with the current best. If the current node is better than the current best, then that node becomes the current best. This process continues until the root node is reached and the search is complete.

For a large number of dimensions, the nearest neighbor algorithm slows down and becomes inefficient. Modifications can be made to adapt to these high

dimensional trees. Keeping track of the best k-nearest neighbors to a point instead of just one can improve the algorithm when high dimensionality is involved. Another method is to approximate the nearest neighbor.

The Best Bin First algorithm approximates a solution for the Nearest Neighbor problem. The algorithm finds the nearest neighbor for a large fraction of queries and finds a very good neighbor the remaining times [24]. A kd-tree with a low dimensionality can use the Nearest Neighbor algorithm. A query using Nearest Neighbor should, with high probability, be within the bin where the query falls, or in a neighboring bin [24]. The algorithm backtracks using a branch-and-bound search. During the backtracking stage, branches of the tree can be thrown away if they represent a space that is further away from the query than current nearest neighbor. In higher dimensionality there are far more bins to be examined. By acepting an approximate nearest neighbor the search can be bounded and return the best nearest neighbor found up to that point.

Approximate match can even be sped up by using multiple randomized kd-trees or searching hierarchical k-means trees with a priority search order [25]. Multiple randomized trees are constructed using the original kd-tree algorithm [4] and splitting the data points in half at each level where the data points show the greatest variance. A single priority queue is maintained across all the trees so the search can be ordered by increasing distance to each bin boundary [25]. This has increased search performance up to about 20 random trees. More than 20 trees leads to no further performance increase or decreases performance.

The hierarchical k-means tree is constructed by splitting data points at each level into *K* distinct regions using k-means clustering [25]. The algorithm is recursively applied to the data points in each dimensional plane until the number of data points is smaller than *K*. This method has shown a performance increase with some datasets but suffers from higher build times than randomized kd-trees.

2.6.1 KD-Tree construction and Matching

The program used for the kd-tree construction and image matching is autopano-sift [5]. The software was developed for panoramic image stitching, which puts together a series of smaller images of a larger scene to yield a single panoramic image. The SIFT keypoints are generated and added to a kd-tree of a large dimensionality (128). The keypoints are loaded from each image in the set and a global kd-tree is created containing all of the keypoints. For every point in the tree the nearest neighbor is approximated using Best Bin First. Matches are grouped into partitions containing at least three matches then filtered using the RANSAC algorithm. Control points are created for the remaining matches. The control points, as seen in Figure 8, are then used to determine what areas in an image matches with another in the partition. The output of this program is a file that can be processed by Hugin [26] to perform panoramic stitching, which goes unused in this research. More importantly, it displays matching features between images and the grouping of images. The groups of images that match to one another are called components.

2.7 Random Sample Consensus

The RANdom SAmple Consensus (RANSAC) algorithm estimates parameters from a set of data which contains outliers [27]. The iterative process randomly selects a subset of data that are hypothetical inliers. The hypothesis is then tested. First a model is fitted to the hypothetical inliers. All of the other data points are tested against the model and if a point fits the model well enough, it is considered a hypothetical inlier. If there are sufficiently many hypothetical inliers then the model is reasonably good. The model is then tested by estimating the error of the inliers relative to the model. This process is repeated a fixed number of times. Every time the model is rejected it is because there are too few points classified as inliers. A new model is accepted if its error is lower than the currently saved model.

2.7.1 RANSAC Model Matching

The RANSAC algorithm filters out inaccurate or incorrect feature matches. For each partition containing similar images a model is fit to determine the geometric consistency of the matches. The incorrect matches are defined by providing a model to the algorithm, which fulfills two things [28]. First, the model can be fit using a small number of input feature matches and secondly, the model can output a fitness value of a novel match once it has been fit. The amount of testing needed, k, is determined by

$$k = w^{-n} + f \cdot SD(k) = w^{-n} + f \cdot \frac{\sqrt{1-w^n}}{w^n}, \tag{12}$$

where w is the fraction of points that is known to be correct, n is the number of elements required to fit the model, SD(k) is the standard deviation of k and f is a chosen factor. The model provides a mapping of coordinate systems between two images [28]. A two-dimensional transformation matrix, M, is used as the model. The matrix fits the model of two image keypoint pairs in two images, I_1 and I_2. That is, there is one line in each image, A_1 to B_1 in I_1 and A_2 to B_2 in I_2. The points A_1 and A_2 are matches, as are B_1 and B_2. These pairs, with a large probability, w, represent the same image feature in both images. The process of coordinate transformation is composed of translating point A_2 into the coordinate origin, rotating the line, by angle α, so that the orientation is the same, scaling of the coordinates, by a scaling factor s, so the two lines are the same length and finally translating A_2 into the position of A_1. The scaling factor is

$$s = \left(\frac{|B_1 - A_1|}{|B_2 - A_2|} \right), \tag{13}$$

and the transformation matrix is

$$M = \begin{bmatrix} s \cdot \cos(\alpha) & s \cdot (-\sin(\alpha)) & s \cdot (\cos(\alpha) \cdot (-A_{2_x}) - \sin(\alpha) \cdot (-A_{2_y})) + A_{1_x} \\ s \cdot \sin(\alpha) & s \cdot \cos(\alpha) & s \cdot (\sin(\alpha) \cdot (-A_{2_x}) + \cos(\alpha) \cdot (-A_{2_y})) + A_{1_y} \\ 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

The position for every keypoint, $P = (x, y)$, in I_2 is used to estimate the expected position in I_1 by multiplying P's homogenous coordinates with the transformation matrix (M)

$$P' = M \cdot P = M \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \quad (15)$$

A comparison can be made between the model expected and the actual keypoint position in I_1. The distance, d, is computed using d(P, M, K) = |K - M • P| = |K - P'|, where K is the actual position of the matching keypoint in I_1 and P' is the expected position. There can be four results from this comparison

- d is small, match is correct. This is the most ideal case.
- d is small, match is incorrect. This case is unlikely, but can occur with repeating elements, such as a series of windows. There are mechanisms in place to make this occurrence unlikely. The quality of the match takes into consideration the distance to the second-best match as well as the best match. If the second best match is a good match, that is two features matching to one feature,then it is likely to be a repeating feature and is discarded [28].
- d is large, match is correct. This occurs with moving objects, where the features are matched correctly but the object has moved within the frame. These cases are discarded.
- d is large, match is incorrect. This case is discarded.

2.8 Summary

The image feature generation algorithms discussed include Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF) and Histogram of

Ordered Gradients (HOG). Each algorithm calculates image features in a particular way. Preprocessing videos or images using high resolution still extraction or critical video quality techniques may increase the usability of smaller videos and images of lesser qualities. With these techniques, lower quality videos and images may produce stronger features to use with matching. Using data structures, such as Bloom filters or kd-trees, to organize keypoint files may lead to an increase in run time and produces more accurate results.

III Methodology

This chapter discusses the specifics of the datasets used in testing and presents the methodology for the matching experiments and how they are measured. Two datasets are used. The first dataset is a series of 125 photos taken from around a home and computer lab. The second dataset are images extracted from the first 5 minutes of a video.

Varied methods are used to calculate metrics for each test. The tests being performed are run time tests, grouping accuracy tests, size and quality comparisons.

3.1 Datasets

Two datasets are tested. The first is from [8] and includes 119 images with resolutions of 1600x1200 and 6 images with resolutions of 640x480 from 6 locations. The locations are a home office, a guest room, a stairwell, a living room, a home exterior and a computer lab. The home office is further split into two groups because the two sets of images taken in the home office are taken 180 degrees off from one another. The images per group is in Table 1. The viewpoints at each location vary widely in rotation, angle and distance from subject. For the interior images the camera distance is between 2.75 feet and 11 feet, the rotation varied by approximately ±15 degrees and the camera angle from the subject varied more than ±50 degrees. The exterior images varied 50 feet with ±10 degrees of rotation and over ±180 cardinal degree direction change.

The second set of images are extracted frames from the first 5 minutes from

the season 13 episode 16 of *Good Eats* entitled "American Classics VII: Don't be a Chicken of Dumplings". The frames were extracted using the ffmpeg program [29]. The first two images from the set were of a black screen and not used because of the absence of any features. The remaining 298 images have a resolution of 656x368. The images contain the upper torsos of one to three people in front of different backgrounds with the exceptions of title screen and two maps. The image groups, classified by the background scene, are the flag, the kitchen, the telephone, the bookshelf, the title screen, the first map and the second map. The breakdown of images per group can be seen in Table 2.

Table 1: Image groups and number of images in each group.

Group	Number of Images
Home Office 1	38
Home Office 2	14
Computer Lab	27
Outside	32
Guest Room	3
Stairwell	3
Living Room	8

3.2 Matching with SIFT

The SIFT software [30] uses the algorithm described in Section 2.1.1. A batch file is used to convert the JPEGs to a Portable Gray Map (PGM) filetype. The SIFT software creates keypoint files from each PGM and a MATLAB® program reduces the keypoints to 102. Another batch file cycles through the keypoint files comparing

Table 2: Image groups and number of images in each group.

Group	Number of Images
Flag	28
Kitchen	136
Telephone	3
Bookshelf	81
Title Screen	10
Map 1	24
Map 2	16

each one to all of the others. The batch file saves the matched points to a text file and creates an image file showing which of the features in the images matched. An example of this can be seen in Figure 7.

3.3 Matching with SURF

The original SURF software, discussed in Section 2.1.2, was implemented by Herbert Bay [31]. The MATLAB® implementation of the SURF matching was implemented by D. Alvaro and J.J. Guerrero. Images are converted to a PGM format and a batch file generates the keypoint files. MATLAB® reduces the keypoints for each file to 102 and performs the matching. Image files similar to the SIFT output are saved showing what features in each image are matched. Text files are also saved with the number of features matched between each image pair.

3.4 Matching with Bloom Filters

The Bloom filter implementation [7] is a simple C program that runs from the

Figure 7 : Sample output from SIFT.

command line. This implementation of a Bloom filter uses two arbitrary hash functions and has a size of 2500000. It parses the input my lines and calculate the hashes of each line. Input is provided by an external file, such as a word list. The Bloom filter was constructed and queries can then be made. If a query was possibly in the Bloom filter then query would be returned and printed to the screen. If the query was not part of the Bloom filter then nothing was printed to the screen. For testing, the keypoint file were provided as input to the Bloom filter.

3.5 Matching with kd-trees

The kd-tree data structure is implemented through autopano-sift [5]. This C

program performs both keypoint file generation and image grouping using two different commands. The kd-tree is constructed during the image group portion of the process. There is an option to downscale the images prior to keypoint generation. The default action is to double the size of the images and then find keypoints. This default option was used in all of the testing.

The matching portion of autopano-sift groups similar images together for panoramic stitching. An example of the output showing the features that match is in Figure 8. The option to turn RANSAC on and off was used for run time testing. For the remaining tests RANSAC was left on and the pre-filter data was taken from the on screen output. The maximum number of matches can also be set. This is the number of matches that will be saved once grouping is done. The default of 16 was left on. Data was again taken from the on screen output for tests that required the total number of matches found.

For the accuracy testing the full dataset is input to autopano-sift while only two images are used as input for size and quality testing.

Figure 8: Sample output from autopano-sift.

3.6 Run Time Metric

The time taken to complete the matching process was measured for each matching method. The times for SIFT and SURF are broken down into step of matching, keypoint generation, keypoint reduction and matching. The kd-tree is broken down into keypoint generation and matching/grouping. The kd-tree does not use reduced keypoints. The kd-tree is run twice, once with RANSAC on and once with it off. This test is run with both datasets.

3.7 Accuracy Calculating

For kd-tree testing the percentage correctly grouped is the proportion of images from the same location that are in the largest component for that location. For example, there are 10 total images of an office. After autopano-sift groups the images, the largest component containing images of the office only contained 8 of them. The percentage correctly grouped together would be 80%, 8 divided by 10. The other 2 images from the office are in their own two separate components, not grouped with any other images.

If the largest group of office images only contained 1 image, meaning the other 9 images are in 9 other components each containing a single office image, then the percentage correctly grouped is 0%. Every image in this case is in a group with just that single image not matched to any other images in the office group. This test is run with both datasets.

3.8 Size and Quality Comparison Metric

Testing for size and quality is performed in the same manner. A subset of 30 images is chosen. For size testing each image is resized to 1600x1200, 800x600, 400x300 and 200x150. The quality of the images stays at 90%.

For quality testing each image is saved at a JPEG quality level of 90%, 70% and 50%. The size of the images stays at 1600x1200.

Three tests are run using each matching method (SIFT, SURF and kd-tree) for size and quality. RANSAC is left on for the kd-tree and the on screen output displays the both the data before RANSAC filtering and after filtering. During testing each size is compared to every other size and each quality is compared to every other quality. The images are only compared to themselves at different sizes and qualities.

The average number of features matched and the standard deviation is calculated. The sizes and qualities are tested to determine if the difference between comparisons and their opposites (quality 90% compared to quality 70% and quality 70% compared to quality 90%) is statistically significant. A two-tailed t-test [32] with a 95% confidence interval is used to determine significance.

3.9 Summary

Four tests are performed using SIFT, SURF and the kd-tree. These test run time, grouping accuracy and the influence of size and quality on the number of features matched. This testing is performed on an Intel Core2 Duo T9500 2.6 GHz laptop with 3.5GB of RAM.

IV Results

This section covers the research results. This includes Bloom filter performance, run time results, grouping accuracy of the kd-tree implementation, size and quality comparison results. These tested the SIFT and SURF algorithms and kd-tree implementation, with and without RANSAC filtering.

4.1 Bloom Filter

Bloom filter software [7] is used to build the filter. The Bloom filter successfully inserted a keypoint file. Trying to query a similar keypoint file did not yield any useful results because of this the Bloom filter was not used for additional testing. The keypoint files are from the Home Office 1 group of 125 dataset.

Table 3 compares the hash values calculated by the Bloom filter. Image 1 and Image 2 are matching keypoints from two similar images. Each row represents a single keypoint that matches using SIFT. The hashes are not similar enough to return a match when a keypoint file is queried.

Hash functions are used to construct Bloom filters. Exact matches are simple to find because they will hash to the same values. Querying for similar matches however becomes difficult. The small differences in similar keypoints result in large changes in the hash values. Bloom filters do not function in a way that would result in matches of similar keypoints.

4.2 Run Time Testing

The timing results for the 125 image set are shown in Table 4. The timing

results of the 298 image set taken from the video are shown in Table 5. The tables show the amount of time it takes to generate the keypoint files for each method, the time it takes to reduce the keypoint files and finally the time it takes to perform the matching. The SIFT algorithm produced, on average, 4474.7 keypoints for the 125 image set and 807.1 for the 298 image set. The SURF algorithm produced, on average, 2442.4 keypoints for the 125 image set and 275.9 for the 298 image set.

The reason that the keypoint generation for the kd-tree software take so much longer is because the autopano-sift software, by default, doubles the size of the images to generate more keypoints. This is resource intensive when starting with large files to begin with.

Even though the second set contains 298 images, the execution times are considerably shorter because the size of the images are smaller (656x368) than the images from the first set (largely 1600x1200). During testing, a set larger than 313

Table 3: Bloom filter hash values.

Hash function 1		Hash function 2	
Image 1	Image 2	Image 1	Image 2
1046594	1732079	48495	2052890
902123	257266	1163211	192032
348358	2482033	770368	2078818
1869278	1439167	342202	1840872
1768528	1835948	1302820	1968954
535535	440890	1566579	1567470
1106069	1716220	1882970	58061
1636090	2481305	671344	56906

images at a resolution of 656x368 exhausts the memory needed for the kd-tree used by the autopano-sift program.

Another factor to consider is that SIFT and SURF matching is run using MATLAB® while autopano-sift grouping is a C implementation. Code optimization may contribute to the faster or slower run times.

4.3 Accuracy

The accuracy results, using the calculation discussed in Section 3.7 are shown in Tables 6 and 7.

The accuracy for the 125 image set using the kd-tree, shown in Table 6, with RANSAC is 51.2% and 73.6% without. Some of the image groups tested are better suited for kd-tree matching. Home Office 1 and Home Office 2 are similar scenes containing the same model of desk with cluttered shelves on opposite sides of the same room. The kd-tree with RANSAC grouped 78.9% of Home Office 1 image together but only 31.7% of Home Office 2. Some of the Home Office 2 images ended

Table 4: Matching time taken for the 125 image set.

Algorithm	Generate Keypoints	Reduce Keypoints	Matching	Total Time Taken	Average # of Keypoints	% of Time Spent Matching
SIFT	23 min	1 hr 36 min	4 hr 5 min	6 hr 4 min	4474.7	67.30%
SURF	3 min	15 min	3 hr 29 min	3 hr 47 min	2442.4	92.10%
KD-tree with RANSAC	2 hr 36 min	N/A	1 hr 11 min	3 hr 47 min	9807.3	31.30%
KD-tree without RANSAC	2 hr 36 min	N/A	1 hr 6 min	3 hr 42 min	9807.3	29.70%

Table 5: Matching time taken for the 298 image/video set.

Algorithm	Generate Keypoints	Reduce Keypoints	Matching	Total Time Taken	Average # of Keypoints	% of Time Spent Matching
SIFT	9 min	39 min	3 hr 47 min	4 hr 35 min	807.1	82.50%
SURF	1 min	28 min	1 hr 29 min	2 hr 1 min	275.9	73.60%
KD-tree with RANSAC	21 min	N/A	29 min	50 min	1072.1	58.00%
KD-tree without RANSAC	21 min	N/A	22 min	43 min	1072.1	51.10%

up being grouped with some of the Home Office 1 images because of the similarities of the scenes. Home Office 1 is also being grouped before Home Office 2 which attempts to construct a full panoramic image of Home Office 1 with the following images from Home Office 2.

RANSAC is spreading the images among too many groups instead of adding image to groups already made. Autopano-sift views each consecutive image as the next part of a panoramic image. With RANSAC on it does not make a connection between office image 1 and office image 5, for example. The first 4 images are similar enough to be grouped together but office image 5 is just different enough to be put in a new group.

The Computer Lab images have a low percentage of correct groupings as well. The Computer Lab heavily featured people that obscured the background scene. The

features that people have and the features that background scenery have are different. The matching that autopano-sift is doing focuses on background features and with the background being behind the people produced a smaller grouping.

The Stairwell images had a large variation in position. These variations were too large for autopano-sift to overcome and match features too. This resulted in each image of the stairwell being in a 3 groups with only 1 image each.

An increase is seen in the largest groups with RANSAC turned off. Without RANSAC to filter away some of the outlying feature autopano-sift becomes less likely to start new groups of images and continue to add to established ones.

The overall accuracy using the kd-tree, shown in Table 7, for the 298 image set is 40.6% with RANSAC and 46.9% without. The largest detriment to grouping this dataset is that the majority of the images feature at least on person. The Kitchen scene had 3 people visibly for some of the images. This is same issued that occurred with

Table 6: Accuracy for 125 image set using the kd-tree with and w/o RANSAC.

Group	Number of Images	w/ RANSAC		w/o RANSAC	
		Largest Component	Percentage Correct	Largest Component	Percentage Correct
Home Office 1	38	30	78.9%	30	78.9%
Home Office 2	14	5	35.7%	12	85.7%
Computer Lab	27	7	25.9%	12	44.4%
Outside	32	16	50%	30	93.8%
Guest Room	3	2	66.7%	3	100%
Stairwell	3	1	0.0%	1	0.0%
Living Room	8	4	50%	5	62.5%
Total			**51.2%**		**73.6%**

the Computer Lab images. A large portion of the background scene is blocked by the people in them. There is also movement around the scene when the host is present so not all parts of the scene are visible the entire time.

A similar problem occurs with the Bookshelf images. This scene is a much closer shot than the Kitchen scene so the host occupies much of the background.

Algorithms for face recognition need to be trained with a large database of faces because these algorithms were not trained it is difficult to identify facial features (eyes, nose, mouth) of the people in the scenes [33]. Clothing are other parts of a person are subjected to the movements of that person. This motion can cause enough change in the feature for it to not be matched. The motion also causes a translation of the feature in the 2D space. The translation cause that feature to by filtered out by quality checking as in Section 2.1.1.2.

The same increase happens with the 298 dataset as with the 125 dataset when RANSAC is off, albeit a smaller increase.

4.4 Size Comparison

Regardless of the order of comparisons, the SIFT algorithm produces the same number of matches for the image size pairs, shown in Table 8.

Half of the time comparing larger images to smaller images produces, on average, more matches than comparing smaller to larger images using the SURF algorithm, shown in Table 9. This can be seen with comparisons of 1-4, 2-4, 3-4 and their opposites.

SURF is not as robust when matching scaled images. The comparisons had a

Table 7: Accuracy for 298 image/video set using kd-tree with and w/o RANSAC.

Group	Number of Images	w/ RANSAC		w/o RANSAC	
		Largest Component	Percentage Correct	Largest Component	Percentage Correct
Flag	28	15	53.6%	17	60.7%
Kitchen	136	50	36.8%	56	41.2%
Telephone	3	2	66.7%	2	66.7%
Bookshelf	81	23	28.4%	29	35.8%
Title Screen	10	8	80%	8	80%
Map 1	24	12	50%	12	50%
Map 2	16	11	68.8%	16	100%
Total			**40.6%**		**46.9%**

difference of about 20 matches for each. As the scale decreases the SURF is unable to extract as many features as with larger images.

Like SIFT, the order of comparisons while using the kd-tree shows no statistically significant differences, with and without RANSAC, shown in Table 10. Significance was calculated using a two-tailed t-test. If the p-value are less than 0.05, which correlates to a 95% confidence interval, then the means are significantly different. Autopano-sift uses SIFT features so it is reason to expect similar results from SIFT and autopano-sift, as is the case for this test. There is also a small increase in average number of keypoints matched with RANSAC off. These extra matches however, may not be good matches. The average matches are rather high in the first place though so the addition of a few extra keypoints matched would not cause the images to be grouped separately.

SIFT is a scale-invariant algorithm. That is, scale does not influence the

features generated from scaled images. SURF is scale-invariant also, but not as robust as SIFT. The number of SURF keypoints detected per octave quickly decays with scale [3].

4.5 Quality Comparison

Like the size comparison tests, SIFT produces the same number of matches regardless of order, shown in Table 11.

The average number of matches found using SURF, shown in Table 12, are very close regarding the order of comparison. Testing shows that the differences in the average number of matches found is not statistically significant.

On average, comparing lower quality image to higher quality, while using the

Table 8: Statistics at different sizes and 90% quality with SIFT.

Size compared to Size	Average Keypoints Matched	Standard Deviation
1 to 2	82.4	4.8
2 to 1	82.4	4.8
1 to 3	65.5	4
3 to 1	65.5	4
1 to 4	56.5	4.9
4 to 1	56.5	4.9
2 to 3	72.1	4
3 to 2	72.1	4
2 to 4	56.1	5.2
4 to 2	56.1	5.2
3 to 4	57.6	6.2
4 to 3	57.6	6.2

Table 9: Statistics at different sizes and 90% quality with SURF.

Size compared to Size	Average Keypoints Matched	Standard Deviation	p value
1 to 2	90.6	3.1	0.62
2 to 1	91.1	4.6	
1 to 3	84.7	4.7	0.63
3 to 1	85.3	3.7	
1 to 4	72.2	5.5	0
4 to 1	52.9	13.2	
2 to 3	88.8	3.5	0.25
3 to 2	87.7	3.8	
2 to 4	73.4	6.4	0
4 to 2	54.6	14.5	
3 to 4	74.5	6.5	0
4 to 3	55.7	12.8	

kd-tree implementation, shown in Table 13, produced more matches. Although, testing shows that these difference are not statistically significant.

All three methods show no statistically significant difference in the average number of keypoints matched. Significance was calculated using a two-tailed t-test. If the p-value are less than 0.05, which correlates to a 95% confidence interval, then the means are significantly different. Again, there is a small increase in the average number of matches for the kd-tree with RANSAC off, but with so many matches already it is difficult to tell if these extra matches are actually meaningful.

Quality did not make much of a difference in the matching/grouping process. The actually difference between 90% quality and 50% quality does no provide enough

Table 10: Statistics at different sizes and 90% quality using kd-tree with and w/o RANSAC.

Size compared to Size	w/ RANSAC			w/o RANSAC		
	Average Keypoints Matched	Standard Deviation	p value	Average Keypoints Matched	Standard Deviation	p value
1 to 2	658	183.1	0.87	661.8	183.9	0.87
2 to 1	665.9	187.1		669.7	187.9	
1 to 3	203.8	51.9	0.95	207	52.6	0.95
3 to 1	204.6	52.1		206.1	52.5	
1 to 4	60.7	17.2	0.18	67.7	18.3	0.94
4 to 1	66.9	18.2		67.4	18.1	
2 to 3	260.5	61.8	0.73	261.8	62	0.87
3 to 2	266.3	65.9		264.4	62.1	
2 to 4	75.7	19.1	0.99	76.8	19.4	0.88
4 to 2	75.7	19		76	19.1	
3 to 4	97.2	22.4	0.95	97.8	22.3	0.96
4 to 3	97.5	21.3		98.1	21.4	

loss in quality for thorough testing. To the human eye, the images appear very close to the same quality.

4.6 Summary

Bloom filters because of the hash functions can not be used to match similar keypoint files. The differences in hash values for similar keypoints are too great to be matched. The matching process is sped up by organizing keypoints into a kd-tree. Autopano-sift does take longer to generate the keypoints because it doubles the size of the images. Grouping accuracy for both datasets experience a decrease when

Table 11: Statistics at different qualities and a 1600x1200 size with SIFT.

Quality compared to Quality	Average Keypoints Matched	Standard Deviation
90%-70%	95.6	3.1
70%-90%	95.6	3.1
90%-50%	93.6	3.1
50%-90%	93.6	3.1
70%-50%	93.5	3.4
50%-70%	93.5	3.4

Table 12: Statistics at different qualities and 1600x1200 size with SURF.

Quality compared to Quality	Average Keypoints Matched	Standard Deviation	p value
90%-70%	100	1.2	0.58
70%-90%	100.2	1.6	
90%-50%	99.3	1.2	0.28
50%-90%	98	6.3	
70%-50%	99.5	1.3	0.4
50%-70%	99.2	1.5	

compared to previous SIFT and SURF matching research. This can be attributed to the background in the scenes being obscured by people. This is seen throughout the 298 dataset results where nearly every scene had at least one person in it.

SIFT and the kd-tree show no significant difference in the average number of keypoints matched. SURF comparisons with the smallest images does show a significant difference. This is due to SURF inability to handle changes in scale as well

Table 13: Statistics at different qualities and 1600x1200 size using the kd-tree with and w/o RANSAC.

Quality compared to Quality	w/ RANSC			w/o RANSAC		
	Average Keypoints Matched	Standard Dev.	p value	Average Keypoints Matched	Standard Dev.	p value
90%-70%	4950.1	1618.7	0.95	4950	1619.2	0.94
70%-90%	4977.7	1610.6		4982.5	1611	
90%-50%	3809	1280.3	0.93	3811.8	1280.2	0.93
50%-90%	3838.4	1282		3842.9	1281.8	
70%-50%	3556.9	1194.9	0.97	3561.3	1194.8	0.97
50%-70%	3544.9	1212		3574.1	1191.3	

as SIFT. All of the methods show no significant in the order of quality matches to a 50% loss of quality.

V Conclusion

This research shows that using a kd-tree implementation does no worse than SIFT and SURF in terms of speed. The accuracy tests showed that the kd-tree implementation grouped the 298 image set with a 40.6% and 46.9% accuracy with RANSAC on and off respectively and grouped the 125 image set with an accuracy of 51.2% and 73.6% with RANSAC on and off respectively. Previous tests showed just the SIFT algorithm matched the 125 image set with an accuracy of 81.6% at its best and 81.1% at its worst. The SURF algorithm had an accuracy of 80.8% at its best and 78.3% at its worst. The autopano-sift program, since it is a panoramic stitcher, assumes the order in which the images are given to the program is the order they are in the panoramic series. This is why the accuracy is much lower using the kd-tree implementation than just the SIFT and SURF algorithms. Some scenes were not ideal for this method of matching because of the addition of people in the scenes.

The order of comparison in regards to size did not make any difference with the SIFT algorithm. The algorithm found the same number of matches regardless if a larger image was being compared to a smaller one or vice versa. The results from testing the order of comparisons using SURF showed that the order mattered for the comparisons with the 200x150 size images. In these cases comparing the larger image to the smaller one produced more matches on average. Using the kd-tree, there was no statistical difference regarding the order of matches, both with RANSAC on and off.

For the quality comparison the SIFT algorithm found the same number of matches for each quality pair. The number of matches found when testing quality

comparisons using SURF algorithm was not statistically significant. The same is true for the matching with the kd-tree, the differences in quality comparison order was not statistically significant.

The kd-tree performs with less accuracy than the stand alone SIFT and SURF algorithms because of it's panoramic nature. In some cases though it was found to perform the matching process faster than the other two algorithms. Keypoint generation takes longer because autopano-sift doubles the size of the images first. This was implemented to extract more keypoints from the images but so many keypoints are found in testing that not using this option may produce similar results with no significant loss to the number of keypoints needed to make correct matches. The difference in quality between images does not effect the number of matches found as much as the difference of size when using SURF and not at all with SIFT. There were large variability with the quality comparison using the kd-tree. Kd-tree implementation uses unreduced keypoint files, producing many more matches than software ultimately uses. Unless explicitly configured otherwise, the software selects the strongest 16 matches by default and uses those.

5.1 Future Work

Further work can be done in experimenting with different data structures to store keypoint files to increase accuracy and speed of matching. Another area to expand this research in is what methods can be used to determine if sections of video have been added, deleted or modified. Applying preprocessing of images as discussed in [15] may be useful. Extract higher quality images from a low quality video to

increase the number of matches found between videos.

Bibliography

1: National Center for Missing and Exploited Children. www.missingkids.com.

2: Lowe, D. G. "Distinctive Image Features from Scale-Invariant Keypoints". *International Journal of Computer Vision*. Vol: 60, Page(s) 91-110. 2004.

3: Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. "Speeded Up Robust Features (SURF)". *Computer Vision and Image Understanding*. Vol: 110, Page(s) 404-417. 2008.

4: Bently, J. L. "Multidimensional Binary Search Trees Used for Associative Searching". *Communications of the ACM*. Vol: 18, Page(s) 509-517. 1975.

5: Nowozin, S. "autopano-sift software". http://user.cs.tu-berlin.de/~nowozin/autopano-sift/index.html.

6: Kirsch, A. and Mitzenmacher, M. "Less Hashing, Same Performance: Building a Better Bloom Filter". *Random Structures & Algorithms*. Vol: 33, Page(s) 456-467. 2008.

7: Hystad, A. "Bloom filter software". http://en.literateprograms.org/Bloom_filter_(C).

8: Fogg, P. N. "Forensic Image Background Matching Using Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF)". Thesis. Air Force Institute of Technology. 2007.

9: Lowe, D. G. "Object Recognition from Local Scale-Invariant Features". *ICCV '99: Proceedings of the International Conference on Computer Vision*. Vol: 2, Page(s) 1150-1158. 1999.

10: Ledwich, L. and Williams, S. "Reduced SIFT Features For Image Retrieval and Indoor Localisation". *Australian Conference on Robotics and Automation*. 2006.

11: Ke, Y. and Sukthankar, R. "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors". *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Page(s) 506-513. 2004.

12: Dalal, N. and Triggs, B. "Histograms of Oriented Gradients for Human Detection". *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol: 1, Page(s) 886-893. 2005.

13: Bensrhair, A., Broggi, A., Rakotomamonjy, A. and Suard, F. "Pedestrian Detection using Infrared images and Histograms of Oriented Gradients". *Intelligent Vehicle Symposium*. Page(s) 206-212. 2006.

14: Dalal, N., Schmid, C. and Triggs, B. "Human Detection using Oriented Histograms of Flow and Appearance". *European Conference on Computer Vision*. Page(s) 428-441. 2006.

15: Chen, D. and Schultz, R.R. "Extraction of High-Resolution Video Stills from MPEG Image Sequences". *Proceedings of the 1998 International Conference on Image Processing*. Page(s) 465-469. 1998.

16: Scultz, R. R. and Steveson, R. L. "Extraction of high-resolution frames from video sequences". *IEEE Transactions on Image Processing*. Page(s) 996-1011. 1996.

17: Korshunov, P. and Ooi, W. T. "Critical video quality for distributed automated video surveillance". *In Proceedings of the 13th Annual ACM international Conference on Multimedia*. Page(s) 151-160. 2005.

18: Low, R. M. and Stamp, M. "Applied Cryptanalysis: Breaking Ciphers in the Real World". Page(s) 193-199. 2007.

19: Chupeau, B., Diehl, E., Lefebvre, F., and Massoudi, A. "Image and Video Fingerprinting: Forensic Applications". *Proceedings of the SPIE*. Vol: 7254, Page(s) 5-14. 2009.

20: Bijhold, J., Geradts, Z., Hermsen, R., and Murtagh, F. "Image Matching Algorithms for Breech Face Marks and Firing Pins in a Database of Spent Cartridge Cases of Firearms". *Forensic Science International*. Page(s) 97-106. 2001.

21: Birchfield, S. "Derivation of Kanade-Lucas-Tomasi Tracking Equation". http://vision.stanford.edu/~birch/klt/derivation.ps. 1996.

22: Gonzalez-Rodriguez, J., Fierrez-Aguilar, J., Ramos-Castro, D. and Ortega-Garcia, J. "Bayesian Analysis of Fingerprint, Face and Signature Evidences with Automatic Biometric Systems". *Forensic Science International*. Vol: 155, Page(s) 126-140. 2005.

23: Maneewongvatana, S. and Mount, D. M. "It's okay to be skinny, if your friends are fat". *4th Annual CGC Workshop on Computational Geometry*. 1999.

24: Beis, J. S. and Lowe, D. "Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces". *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*. Page(s) 1000-1007. 1997.

25: Lowe, D. G. and Muja, M. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration". *VISAPP International Conference on Computer Vision Theory and Applications*. Page(s) 331-340. 2009.

26: D'Angelo, P. "Hugin software". http://hugin.sourceforge.net/.

27: Bolles, R. C. and Fischler, M. A. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Communications of the ACM*. Vol: 24, Page(s) 381-395. 1981.

28: Nowozin, S. "Image Align Model Used in autopano-sift". *http://user.cs.tu-berlin.de/~nowozin/autopano-sift/matchmodel.pdf*. 2004.

29: Bellard, F. "ffmpeg software". http://www.ffmpeg.org/.

30: Hess, R. "SIFT software". http://web.engr.oregonstate.edu/hess.

31: Bay, H., Van Gool, L. and Tuytelaars, T. "SURF software". http://www.vision.ee.ethz.ch/surf/index.html.

32: Student. "The Probable Error of a Mean". *Biometrika*. Vol: 6, Page(s) 1-25. 1908.

33: Chellappa, R., Phillips, P. J., Rosenfeld, A., and Zhao, W. "Face recognition: A literature survey". *ACM Computing Surveys*. Vol: 35, Page(s) 399-458. 2003.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
16-09-2010	**Master's Thesis**	Sep 2008 – Sep 2010

4. TITLE AND SUBTITLE

Applying Image Matching to Video Analysis

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Adam Jay Behring

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCO/ENG/10-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Multi-Sensor Exploitation Branch
Mr. Chad Heitzenrater
525 Brooks Rd.
Rome, NY 13441
(315)330-2575
Chad.Heitzenrater@rl.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RIEG

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Dealing with the volume of multimedia collected on a daily basis for intelligence gathering and digital forensics investigations requires significant manual analysis. A component of this problem is that a video may be reanalyzed that has already been analyzed. Identifying duplicate video sequences is difficult due to differences in videos of varying quality and size. This research uses a kd-tree structure to increase image matching speed. Keypoints are generated and added to a kd-tree of a large dimensionality (128 dimensions). All of the keypoints for the set of images are used to construct a global kd-tree, which allows nearest neighbor searches and speeds up image matching. The kd-tree performed matching of a 125 image set 1.6 times faster than Scale Invariant Feature Transform (SIFT). Images were matched in the same time as Speeded Up Robust Features (SURF). For a 298 image set, the kd-tree with RANSAC performed 5.5 times faster compared to SIFT and 2.42 times faster than SURF. Without RANSAC the kd-tree performed 6.4 times faster than SIFT and 2.8 times faster than SURF. The order images are compared to the same images of different qualities, did not produce significantly more matches when a higher quality image is compared to a lower quality one or vice versa. Size comparisons varied much more than the quality comparisons, suggesting size has a greater influence on matching than quality.

15. SUBJECT TERMS
image matching, video matching, SIFT, SURF, kd-tree

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Dr. Gilbert L. Peterson
U	**U**	**U**	**UU**	64	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 2785-6565, ext 4281; e-mail: Gilbert.Peterson@afit.edu